#17

10/21/03

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re appl. of: **Hooper et al.** § Group Art Unit: **2142**
§
Serial No.: **09/213,907** § Examiner: **Vu, T.**
§
Filed: **12/17/1998** § Atty. Docket No.: **AT9-98-561**
§

For: **Manager Object for Management of Multiple Resources on Dataless Clients in a Distributed Computing Environment**

Certificate of Mailing
Under 37 C.F.R. § 1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to:
Mail Stop Appeal Brief--Patent
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450
on October 4, 2003.

By: _____
Joseph R. Burwell, Reg. No 44,468

APPELLANT'S BRIEF
IN RESPONSE TO OFFICE ACTION UNDER 37 C.F.R. § 1.192

This brief is filed in triplicate in support of the Notice of Appeal, filed 03/20/2003, and which appealed from the decision of the examiner dated 11/20/2002 rejecting claims 1-23. The fee required under 37 C.F.R. § 1.17(c) for filing a brief in support of an appeal was paid with the first submission of the appeal brief, which was filed 06/20/2003.

This second submission of the appeal brief contains a new summary of the invention and is filed in response to the Notice of Non-Compliance, mailed 09/23/2003, based on the examiner's objection to the summary of the invention within the first submission of the appeal brief for not including references to pages and lines in the specification and reference numerals in the drawings.

1. REAL PARTY IN INTEREST

The real party in interest in this appeal is International Business Machines Corporation (IBM).

2. RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

3. STATUS OF CLAIMS

Claims 1-23 are pending in this application; claims 1-23 have been finally rejected; claims 1-23 have been appealed. No claims have been canceled, withdrawn, or allowed.

4. STATUS OF AMENDMENTS

No after-final amendments have been filed.

5. SUMMARY OF INVENTION

The present invention is a method for managing a given application type and, in particular, all instances of that application type. A manager object or entity **36a-36c** (specification--page 14, line 12) is created to represent each application type running on a client **30** (specification--page 14, line 8). The manager object may reside anywhere in the distributed network **10** (specification--page 9, line 3), and the manager object is the target of management operations, which are then redirected to the appropriate client node **18** or **19** (specification--page 9, line 17). The distributed data processing system preferably includes a dataless management framework, also called a lightweight client framework (LCF) **24** (specification--page 11, line 13). One advantage of the present invention is that a server management operation may be directed to a particular type or a given application instance without exposing the remainder of the client nodes to the operation. The manager object preferably comprises a control routine **40** (specification--page 14, line 16) and a registry **35** (specification--page 14, line 15) containing a set of one or more elements **38a-38b** (specification--page 14, line 18) in which each element is a dataset of context-specific information for a given application instance. The context information may include client node identity, installation location, e.g., a directory, installation identifier, e.g., database server name, or other administrative details. When a management operation, such as a query, is performed by a management server, the manager object redirects the query to the client node after possibly augmenting it with appropriate

context information (step **62** at specification--page 18, line 1). Upon receipt of the query at the client (step **64** at specification--page 18, line 3), a query agent is started (step **66** at specification--page 18, line 3), and the context information is then used by a local query agent to identify which of the many installed instances of the application to target for the management operation.

6. ISSUES

The issues on appeal are:

whether claims 1, 4-10, 15-17, and 19-23 are unpatentable over by Khoyi et al., "Matchmaker for Assisting and Executing the Providing and Conversion of Data Between Objects in a Data Processing System Storing Data in Typed Objects Having Different Data Formats", U.S. Patent No. 5,261,080, filed 08/28/1992, issued 11/09/1993 (hereinafter Khoyi et al.), in view of Jeffords et al., "Replicated Resource Management System for Managing Resources in a Distributed Application and Maintaining a Relativistic View of State", U.S. Patent No. 6,233,623, filed 01/11/1996, issued 05/15/2001 (hereinafter Jeffords et al.); and

whether claims 2, 3, 11, 12, and 18 are unpatentable over Khoyi et al. in view of Jeffords et al. and further in view of Bereiter, "Software Auditing Mechanism for a Distributed Computer Enterprise Environment", filed 10/01/1996, issued 05/19/1998.

It should be noted that Appellant noted in two responses to Office actions that the Office actions failed to recite some of the dependent claims in any statement of grounds of

rejection; specifically, in the final Office action, these claims were claims 4-9, 13-16, and 19-21. More importantly, in the final Office action, dependent claims 13 and 14 fail to appear in any of the arguments for the rejections while also failing to appear in the statement of the grounds of rejection. Hence, it seems possible that dependent claims 13 and 14 may be allowable in the judgment of the PTO. However, Appellant has placed claims 13 and 14 in Group A.

7. GROUPING OF CLAIMS

The claims stand and fall together as follows:

Group A -- claims 1, 4-6, 9, 10, 13, 14, 16, 19-21, and 23;

Group B -- claims 17 and 22; and

Group C -- claims 7, 8, and 15.

8. ARGUMENTS

Argument 8.A.

Was 35 U.S.C. § 103(a) properly applied in a rejection of claims 1, 4-6, 9, 10, 13, 14, 16, 19-21, and 23 (Group A) as being unpatentable over Khoyi et al. in view of Jeffords et al.?

Arguments in support of patentability

With respect to the grouping of claims for arguments in support of patentability, independent claims 1 and 10 are method claims, and independent claim 23 is directed to a software framework. Although these independent claims are similar, claim 1 is broader than the other independent claims.

Hence, for purposes of this argument, Appellant argues for the patentability of the present invention using claim 1 as an exemplary claim.

Rejections under 35 U.S.C. 103 must provide a *prima facie* case for obviousness. According to 37 C.F.R. § 1.192(c)(8)(iv), for each rejection under 35 U.S.C. § 103, Appellant must specify the errors in the rejection, the specific limitations in the rejected claims which are not described in the prior art relied on in the rejection, and how such limitations render the claimed subject matter nonobvious over the prior art. If the rejection is based upon a combination of references, the argument shall explain why the references, taken as a whole, do not suggest the claimed subject matter. In summary, Appellant argues that the pending claims in the present patent application are patentable because the rejection of the independent claim 1 fails to provide a *prima facie* case of obviousness. Independent claim 1 reads:

> 1. A method of managing a set of clients in a distributed computer network having a management server, comprising the steps of:
>
> associating a manager object to each application type on a given client, the manager object including a registry having a set of one or more elements, wherein each element includes information representing a context of an application instance; and
>
> managing all instances of the application through the manager object.

Rejection of claim 1

The rejection of claim 1 reads as follows in its entirety:

> As per claims 1, 10, 17, 22, 23 Khoyi discloses a method of managing a set of clients in a distributed

computer network having a management server, comprising the steps of:

associating a manager object to each application type on a given client (col 9 lines 38-col 10 line 29, col 13 lines 30-52]; and

managing all instances of the application through the manager object which is equivalent to the object managers are peer with each other and are children of the application manager [col 14 lines 40-45]. Khoyi also taught the object catalog for a link registry [col 70 lines 15-32]. However Khoyi is silent on the manager object including a registry having a set of one or more elements wherein each element includes information representing a context of an application instance. A skilled artisan would have motivation to improve the manager object on Khoyi's apparatus and found Jeffords teaching. Jeffords discloses a manager object including a registry having a set of one or more elements such as manager object having an associated set of memory pools and a registry of the network unique identifiers for the resource objects in the associated set of memory pools where each pool representing an RRM process which is equivalent to the element includes information representing a context of an application instance [col 10 lines 30-43, col 11 lines 20-62]. Khoyi-Jeffords also taught redirecting the modified query to the client machine [Khoyi col 43 lines 33-55] [Jeffords col 15 lines 47-67].

Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to incorporate the registry with a set of memory pools as taught by Jeffords into the Khoyi's apparatus in order to improve the manager object. Doing so would provide a simple, dynamic and efficient process to the manager object of the distributed computing system.

Thus, the system and method of claims 1, 10, 17, 22 and 23 is obvious in view of the combination of references.

Analysis of the rejection

The rejection of the claim 1 under Khoyi et al. and Jeffords et al. starts with an attempt to build a foundation

for an obviousness argument with multiple citations to passages of Khoyi et al.; these passages are applied against a portion of the first element of claim 1. **However, none of these passages disclose the claimed features of the independent claims.** When the rejection turns to Jeffords et al. as a secondary reference for disclosure of other claim elements that are admittedly not shown in Khoyi et al., Jeffords et al. suffers from the same deficiencies, i.e. none of these passages disclose the claimed features of the independent claims.

More specifically, the rejection merely applies portions of Khoyi et al. and Jeffords et al. against claim elements without careful consideration of the claim language and the subject matter. Passages from the references appear to have been cited in the rejection only because the passages included terminology that seemed similar to the claim language. The form of the rejection appears appropriate, but the argument presented by the rejection is illogical because the basis of the rejection, i.e. the cited portions of the reference, do not disclose what the rejection purports that they disclose. More importantly, the references fail to disclose the claimed elements of the present invention.

The rejection addresses the first portion of the first element of independent claim 1 with reference to two portions of Khoyi et al., which together provide a sufficient description of the operation of the disclosed system and state the following (emphasis added):

> The system of the present invention is a member of the general class of systems described as "object based". That is, information is stored in structures referred to

as "objects". In the implementation of the present preferred embodiment most objects each correspond to one or more files of a conventional computer file system.

Further with regard to objects, the system of the present invention allows the use of an essentially unlimited variety of object "types", including the type previously described as a folder, wherein there may be a type of object for each form of data or information or operation to be performed by the system. That is, the system of the present invention defines only the minimal interface between an object and the system and does not define the internal structure or form of any object. As such, an object within the system of the present invention may be regarded as a general purpose container for data, programs or other information, with the internal structure or form of a particular object being defined by the requirements of the operation to be performed or the type of data or information to be stored therein.

Programs for operating upon objects are known as "object managers" or are sometimes referred to as "editors", "applications programs", or "applications". The term "application" is also used to refer to a collection of object managers that operate on a single object type. Each type of object has associated with it at least one object manager that is designed or intended as the primary means for operating upon the data or information stored in that type of object. For example, the system may support a "document type" object for word processing and there will be a word processing object manager associated with that object type. Similarly, a "data base type" object will have associated with it a data base object manager, which is the primary means for operating upon or with the data stored in the data base type object.

It should be noted, however, that the object managers for operating with a particular type of object are not limited to the primary object manager for that object type. Moreover, the particular object manager invoked to operate upon a particular object may depend upon the type of operation to be performed. The system of the present invention provides for a plurality of object managers to operate with any given object type and certain object managers, for example, certain utilities,

may operate with more than one type of object. The primary object manager is simply the object manager which is invoked by default for operations upon a particular object type if the user does not select a different program.

Although typically an object manager will operate on the data of a single type of object, in certain cases it may be desirable to arrange a program to be an object manager for more than one object type. Further, there are various utility programs that perform operations that do not interpret object data (e.g. file copy) and that therefore can be used with various types of objects. --(Khoyi et al., col. 9, line 38, to col. 10, line 29).

For each type of object there is one or more "object manager" that can operate on objects of that type. **Object managers roughly correspond to applications programs.** For example, there may be spreadsheet program: this can be understood to be an "application", as distinguished from operating system program; it can also be understood to be a "manager" or "editor" of objects of type "spreadsheet".

Because it is an object's object manager(s) that define its structure and interpretation, the collection of object types is open-ended; existing types of information can be integrated with future types of information with the same each with which the existing types are integrated with each other. A new object type can be added to the system by adding a program (i.e., an object manager) that can manipulate objects of the new type. The ability to manipulate the new type of object need only be implemented once. Having added the new object manager, objects of the new type can be linked into, exchange data with, and otherwise appear integrated with objects of pre existing types without modification to pre existing object managers. --(Khoyi et al., column 13, lines 30-52).

In the system of Khoyi et al., an object manager manages objects of a particular object type. As clearly shown in the

recited portions of Khoyi et al., **object managers correspond to application programs**.

This set of features is not similar to the claim elements against which they are cited. In the present invention, a manager object is associated with an application type, and the manager object manages instances of the application. Hence, manager objects are distinct entities from instances of applications.

The rejection is confusingly silent on the correspondence between features in the system of Khoyi et al. and the present invention. The rejection merely cites a section of Khoyi et al. without taking a position on the which entities in the system of Khoyi et al. might be equivalent to entities in the present invention. Appellant conjectures that the position of the rejection is that an object manager in the system of Khoyi et al. is similar to a manager object in the present invention.

However, if this is the case, then there are no entities in the system of Khoyi et al. that correspond to instances of applications in the present invention because an object manager in the system of Khoyi et al. corresponds to an application. An object manager in the system of Khoyi et al. cannot correspond to a manager object in the present invention, which manages instances of applications in the present invention, and also correspond to an instance of an application, as stated within Khoyi et al. itself. In other words, an object manager in the system of Khoyi et al. cannot be both an entity that manages instances of applications and the entities that are being managed, i.e. both the managing

entity and the managed entity, yet this is the illogical position that seems to be asserted by the rejection.

The rejection then addresses the second element of independent claim 1 with reference to a portion of Khoyi et al., which states the following:

> In the present application, there is an Application Manager process. The Application Manager spawns the object manager processes. Thus, in general, the object managers are peers with each other and are children of the Application Manager.
> --(Khoyi et al., column 14, lines 40-45).

The only way that this section of Khoyi et al. could be logically interpreted as being relevant to the claim language is that the Application Manager in the system of Khoyi et al. is somehow analogous to a manager object in the present invention. However, the rejection has already taken a position as to which entity in the system of Khoyi et al. is supposedly equivalent to a manager object in the present invention. Hence, the rejection appears to assert contradicting positions.

Moreover, the Application Manager does not have the characteristic of being associated with any one particular application type; the Application Manager manages all object managers in the system of Khoyi et al.. Hence, an assertion that the Application Manager is supposedly equivalent to a manager object in the present invention also fails with respect to the claim language because the claim states: "associating a manager object to each application type".

Contrary to the statements in the rejection, Khoyi et al. fails to show the first portion of the first element of claim

1 and the second element of claim 1. The rejection then turns to Jeffords et al. and states that Jeffords et al. discloses the second portion of the first element of claim 1, i.e. "the manager object including a registry having a set of one or more elements, wherein each element includes information representing a context of an application instance". The rejection asserts that Jeffords et al. teaches this feature in two sections of the reference, which recited hereinbelow. However, these sections in Jeffords et al. are recited not for ease of reference for the reader but to support Appellant's contention that these sections are irrelevant to the claimed feature.

> Replication of an object's construction occurs when an object is first created within a resource pool, and when a resource pool is synchronized. Replication of an object's attributes occurs incrementally as the attributes' state changes. Both of these replication functions require an object transport function.
>
> The RRM provides a means by which software objects are transported between processes. The object transport function is able to take any "supported" software object and convert it to a transportable (distributable) representation that may be sent to and received from any RRM processes. Supported software objects are objects that can be distributed (streamed).
>
> Thus, the object transport function obtains an object's distributable representation from that object and creates an object from its distributable representation; this is accomplished by an object packetization service, also known as streaming and unstreaming, or packing and unpacking.
> --(Jeffords et al., column 10, lines 30-43).

> When an RRM process enters a distributed computing function it must send a join pool request to all active RRM processes for each resource pool it is interested in. Each one of the active processes must respond with either a positive acknowledgment (meaning it is also interested

in the pool specified), or a negative acknowledgment (it is not interested). Each of the positively responding processes must then send all of the objects it is responsible for ("owns") in the pool to the process joining the pool (when a pure in memory replication scheme is used). If the process joining the pool already has objects in that pool that it is responsible for, it must send those objects to each of the processes that responded positively. In this way, each pool is synchronized in both directions for every process that is interested in the pool.

A separate level of synchronization and consistency checks occurs at the memory level. This ensures that: (1) a consistent view of memory pool existence and RRM process attachment is maintained (i.e., what processes have joined what pools); (2) each pool's objects are distributed and maintained consistently; and (3) the relative state of pool synchronization is distributed and checked.

Memory level consistency checks include:
agreement as to which RRM processes are interested in each pool; and
relativistic synchronization of each pool.

Memory level synchronization includes:
distribution of all objects created within a pool to all RRM processes interested in (joined to) the pool (synchronization of object existence); and
bidirectional pool synchronization and agreement upon the pool's synchronization state between all RRM processes joined to each pool.

Memory level synchronization is achieved when:
every resource pool an RRM process is interested in is synchronized with respect to the given RRM process; a resource pool is synchronized with respect to a given RRM process when all other RRM processes have sent their owned objects to the given RRM process and the given RRM process has sent its owned objects to all other interested RRM processes.

--(Jeffords et al., column 11, lines 20-62).

It is entirely unclear why these particular sections of Jeffords et al. are referenced by the rejection.

The rejection states that Jeffords et al. discloses "a manager object having an associated set of memory pools and a registry of the network unique identifiers for the resource objects in the associated set of memory pools where each pool representing an RRM process". Elsewhere, Jeffords et al. states the following, which might seem to contain the feature that is stated within the rejection:

> [E]ach RRM [Replicated Resource Management] process has an associated resource manager (RM) which identifies several resource pools of interest, i.e., several meetings which are of interest to the person. In this way, a resource manager (process) participates in just its areas (pools) of interest, maintains a registry of resources in these associated pools, and includes mechanisms for communicating with other resource managers (processes) in order to maintain a consistent view of the state of the resources in the associated pools. --(Jeffords et al., column 2, lines 32-41).

However, Appellant asserts that the feature of resource pools or memory pools in the system of Jeffords et al. is not equivalent to the claim element against which this feature is compared, namely, "the manager object including a registry having a set of one or more elements, wherein each element includes information representing a context of an application instance". More importantly, the feature of a context of an application instance is not present at all in Jeffords et al..

As shown by Appellant in the arguments above, Khoyi et al. and Jeffords et al. do not disclose the claim elements as stated in the rejection. The rejection then proceeds to state that it would have been obvious to combine features from the

system disclosed in Khoyi et al. and the system disclosed in Jeffords et al.. However, the motivational statement in the rejection is completely generic: "... in order to improve the manager object. Doing so would provide a simple, dynamic and efficient process to the manager object of the distributed computing system." The rejection does not provide any argument as to how or why a characteristic of the system of Khoyi et al. or a characteristic of the system of Jeffords et al. would motivate one of ordinary skill in the art to combine any feature of these systems with any feature of the other system. Appellant admits that, in general, one of ordinary skill in the art would desire an "efficient process" and would want to "improve" a system. However, Appellant asserts that the rejection completely fails to provide a reason why one would be motivated to combine features from the prior art references.

Jeffords et al. does not disclose the features that Khoyi et al. fails to disclose, which were discussed above. Therefore, both Khoyi et al. and Jeffords et al. both fail to disclose the claimed elements of claim 1. In addition, there is no motivation or suggestion in the prior art for combining features from these references, and even if one were motivated to do so, the features of the system that is disclosed in Jeffords et al. could not be integrated into the system that is disclosed in Khoyi et al. to form the claimed features that are not shown in Khoyi et al. as asserted by the rejection.

Examiner bears the burden of establishing a *prima facie* case of obviousness

Contrary to the assertions of the rejection, Khoyi et al. clearly fails to show claimed features of the present invention. Moreover, both Khoyi et al. and the combination of Jeffords et al. and Khoyi et al. fail to show the claimed features. As should be recognized, because both the primary and secondary references in the rejection fail to disclose the claimed features against which the references were applied, and because the references fail to be combinable to produce these features, the rejection fails to fulfill the requirements of a proper obviousness argument.

Hence, a rejection of claim 1 cannot be based upon the cited prior art to establish a *prima facie* case of obviousness. Therefore, a rejection of the claim 1 under 35 U.S.C. § 103(a) has been shown to be insupportable in view of the cited prior art, and claim 1 is patentable over the applied references. For this and other reasons, Appellant argues that the position of the Examiner should be reversed and that the rejection of claim 1 should not be upheld.

Argument 8.B.

Was 35 U.S.C. § 103(a) properly applied in a rejection of claims 17 and 22 (Group B) as being unpatentable over Khoyi et al. in view of Jeffords et al.?

Arguments in support of patentability

With respect to the grouping of claims for arguments in support of patentability, independent claim 17 is directed to

a manager object, and independent claim 22 is directed to a computer program product. Although these independent claims are similar, claim 22 is broader than independent claim 17 because claim 17 also includes a registry. Hence, for purposes of this argument, Appellant argues for the patentability of the subject matter using claim 22 as an exemplary claim. In summary, Appellant argues that the pending claims in the present patent application are patentable because the rejection of the independent claim 22 fails to provide a *prima facie* case of obviousness. Independent claim 22 reads:

> 22. A computer program product in a computer-readable medium and executable on a given computer for use in managing an application type executing on a client machine, comprising:
>
> means for intercepting a query directed to the client machine for managing the application type;
>
> means for modifying the query with information representing a context of a given application instance identified in the query; and
>
> means for redirecting the modified query to the client machine to target management of the given application instance directly.

The rejection of the independent claims generally states: "Khoyi-Jeffords also taught redirecting the modified query to the client machine [Khoyi col 43 lines 33-55] [Jeffords col 15 lines 47-67]." This statement apparently addresses the third element in claim 22. The section of Khoyi et al. that is cited by the rejection is recited below; again, this section is recited not for ease of reference for the reader but to support Appellant's contention that this section is irrelevant to the claimed feature.

> When the source data is similar to or identical to the destination data, either directly or after

> conversion, the copied, moved or shared source data will "blend into" or become "embedded" in the destination data. In the case of copied or moved data, the copied or moved data may be editable by the destination object manager. In the case of shared data, however, and while the shared data may be copied, moved or deleted, it may be edited only in the source object and only by an object manager for the source object. The destination object's object manager may convert the shared data into its own internal data format but may not incorporate the shared data into its own data, may not edit the source data in any way, and is required to "mark and hold" the shared data. The data should be visually marked for the user, so the user is aware that the data is shared, rather than directly a part of the object being edited. In the case of data that is "marked and held", an entry in the link parallel file of the destination object will contain (i.e., "hold") a copy of the linked data. This copy in the link parallel file cannot directly be edited: rather, this data is edited by editing the source object from which it is linked.
> --(Khoyi et al., col. 43, lines 33-55).

It is entirely unclear why this particular section of Khoyi et al. has been cited by the rejection as disclosing a feature concerning "queries"; it is clear that this particular section is completely irrelevant to the claimed feature.

The section of Jeffords et al. that is cited by the rejection against the feature in independent claim 22 is recited below; again, this section is recited not for ease of reference for the reader but to support Appellant's contention that this section is irrelevant to the claimed feature.

> The embodiment described herein assumes a communications network is modelled in memory, such as cache memory in a network manager. The network manager may be a Sun workstation sold by Sun Gateway 2000 with a Sun Solaris Microsoft Windows NT operating system; the network management software may be the Spectrum.TM. Network Management software sold by Cabletron Systems,

> Inc., Rochester, N.H., USA and described in U.S. Pat. No. 5,261,044 by R. Dev et al., which is hereby incorporated by reference in its entirety. Typically, the network manager resides in a host and the resources of a network provide alerts to the host which are received and interpreted by the network manager in accordance with known techniques. The resources of a network may include a plurality of individual workstations, personal computers, modems, applications resident in host computers, communication satellites, and the like. In a modeled network, a change made by an application to one resource of the network, which affects the other resources of the networks, will be immediately known and accounted for since the model of the network within the data cache manager accounts for these inter-relationships.

Again, it is entirely unclear why this particular section of Jeffords et al. has been cited by the rejection as disclosing a feature concerning "queries"; it is clear that this particular section is completely irrelevant to the claimed feature.

Jeffords et al. does not disclose the features that Khoyi et al. fails to disclose. Therefore, both Khoyi et al. and Jeffords et al. both fail to disclose the claimed elements of claim 22. In addition, there is no motivation or suggestion in the prior art for combining features from these references, and even if one were motivated to do so, the features of the system that is disclosed in Jeffords et al. could not be integrated into the system that is disclosed in Khoyi et al. to form the claimed features that are not shown in Khoyi et al. as asserted by the rejection.

Contrary to the assertions of the rejection, Khoyi et al. clearly fails to show claimed features of the present invention. Moreover, both Khoyi et al. and the combination of

Jeffords et al. and Khoyi et al. fail to show the claimed features. As should be recognized, because both the primary and secondary references in the rejection fail to disclose the claimed features against which the references were applied, and because the references fail to be combinable to produce these features, the rejection fails to fulfill the requirements of a proper obviousness argument.

Hence, a rejection of claim 22 cannot be based upon the cited prior art to establish a *prima facie* case of obviousness. Therefore, a rejection of the claim 22 under 35 U.S.C. § 103(a) has been shown to be insupportable in view of the cited prior art, and claim 22 is patentable over the applied references. For this and other reasons, Appellant argues that the position of the Examiner should be reversed and that the rejection of claim 22 should not be upheld.

Argument 8.C.

Was 35 U.S.C. § 103(a) properly applied in a rejection of claims 7, 8, and 15 (Group C) as being unpatentable over Khoyi et al. in view of Jeffords et al.?

Arguments in support of patentability

Claims 7, 8, and 15 are related to a step of discovering an application type. For purposes of this argument, Appellant argues for the patentability of the subject matter using claim 7 as an exemplary claim. Dependent claim 7 reads:

> 7. The method as described in claim 1 wherein the application type is discovered by the manager object.

The rejection of claim 7 states that claim 7 is shown in Jeffords et al. by an RRM discovery process. However, Jeffords et al. states: "During this discovery process, each RRM process learns about the relative state of all other RRM processes in the network."--(Jeffords et al., col. 5, lines 22-24). As already discussed above, the rejection argued that Khoyi et al. disclosed something about application types, so it is unclear how Jeffords et al. now discloses something about discovering an application type that was supposedly already found in the primary reference. More importantly, this passage from Jeffords et al. clearly does not disclose the claim element, which requires "the application type is discovered by the manager object".

Contrary to the assertions of the rejection, Jeffords et al. clearly fails to show the features of claim 7. Moreover, both Khoyi et al. and the combination of Jeffords et al. and Khoyi et al. fail to show the claimed features. As should be recognized, because both the primary and secondary references in the rejection fail to disclose the claimed feature against which the references were applied, and because the references fail to be combinable to produce this feature, the rejection fails to fulfill the requirements of a proper obviousness argument.

Hence, a rejection of claim 7 cannot be based upon the cited prior art to establish a *prima facie* case of obviousness. Therefore, a rejection of the claim 7 under 35 U.S.C. § 103(a) has been shown to be insupportable in view of the cited prior art, and claim 7 is patentable over the applied references. For this and other reasons, Appellant argues that

the position of the Examiner should be reversed and that the rejection of claim 7 should not be upheld.

9. Conclusion

In view of the above arguments, it is respectfully urged that the rejections of the claims should not be sustained.

DATE: October 4, 2003

Respectfully submitted,

Joseph R. Burwell
Reg. No. 44,468
ATTORNEY FOR APPELLANT

Law Office of Joseph R. Burwell
P.O. Box 28022
Austin, Texas 78755-8022
Voice: 866-728-3688 (866-PATENT8)
Fax: 866-728-3680 (866-PATENT0)
Email: joe@burwell.biz

10. APPENDIX OF CLAIMS

1. A method of managing a set of clients in a distributed computer network having a management server, comprising the steps of:

associating a manager object to each application type on a given client, the manager object including a registry having a set of one or more elements, wherein each element includes information representing a context of an application instance; and

managing all instances of the application through the manager object.

2. The method as described in claim 1 wherein the given client supports a dataless management framework.

3. The method as described in claim 1 wherein the dataless management framework includes a local agent that is controlled by the manager object to manage the application instance.

4. The method as described in claim 1 wherein the element includes information identifying a client node.

5. The method as described in claim 1 wherein the element includes information identifying a directory where the application instance is installed.

6. The method as described in claim 1 wherein the element includes information identifying a name of a resource where the application instance is installed.

7. The method as described in claim 1 wherein the application type is discovered by the manager object.

8. The method as described in claim 1 further including the step of discovering the application type prior to associating the manager object.

9. The method as described in claim 1 wherein the manager object is managed by the management server.

10. A method of managing a set of clients in a distributed computer network having a management server, comprising the steps of:

for each client, establishing a set of manager objects, wherein each manager object is associated with a given type of application to be managed;

responsive to management operations initiated at the management server, managing a given client using the set of manager objects established for that client.

11. The method as described in claim 10 wherein the given client supports a dataless management framework.

12. The method as described in claim 11 wherein the dataless management framework includes a local agent that is controlled by the set of manager objects for the given object.

13. The method as described in claim 10 wherein each manager object includes a registry comprising a set of elements, wherein each element includes information representing a context of an application instance.

14. The method as described in claim 13 wherein the information comprises a client node identity, an installation location, and an installation identifier.

15. The method as described in claim 10 further including the step of discovering the application type.

16. The method as described in claim 10 further including the step of registering the application type with the manager object.

17. A manager for use in managing an application type, the manager executing on a client machine, comprising:

a registry comprising a set of one or more data elements, each element including information representing a context of an application instance; and

a control routine (a) for intercepting a query directed to the client machine, (b) for modifying the query as a function of the information; and (c) for redirecting the modified query to the client machine to target management of the application instance.

18. The manager object as described in claim 17 further including a routine for discovering application types executing on the client machine.

19. The manager object as described in claim 17 wherein the information identifies a client node.

20. The manager object as described in claim 17 wherein the information identifies a directory where the application instance is installed.

21. The manager object as described in claim 17 wherein the information identifies a name of a resource where the application instance is installed.

22. A computer program product in a computer-readable medium and executable on a given computer for use in managing an application type executing on a client machine, comprising:

means for intercepting a query directed to the client machine for managing the application type;

means for modifying the query with information representing a context of a given application instance identified in the query; and

means for redirecting the modified query to the client machine to target management of the given application instance directly.

23. A framework for managing a set of clients, comprising:

a management server to which the set of clients are connectable; and

for each client, a set of manager objects, wherein each manager object is associated with a given type of an application to be managed at the client and is responsive to management operations initiated at the management server for managing one or more application instances directly.